Numeric and symbolic evaluation of the pfaffian of general skew-symmetric matrices

C. González-Ballestero[a], L.M. Robledo[a,*], G. F. Bertsch[b]

[a] *Departamento de Física Teórica, Universidad Autónoma de Madrid, E-28049 Madrid, Spain*
[b] *Department of Physics and Institute for Nuclear Theory, University of Washington, Seattle, WA 98195–1560 USA*

Abstract

Evaluation of pfaffians arises in a number of physics applications, and for some of them a direct method is preferable to using the determinantal formula. We discuss two methods for the numerical evaluation of pfaffians. The first is tridiagonalization based on Householder transformations. The main advantage of this method is its numerical stability that makes unnecessary the implementation of a pivoting strategy. The second method considered is based on Aitken's block diagonalization formula. It yields to a kind of LU (similar to Cholesky's factorization) decomposition (under congruence) of arbitrary skew-symmetric matrices that is well suited both for the numeric and symbolic evaluations of the pfaffian. Fortran subroutines (FORTRAN 77 and 90) implementing both methods are given. We also provide simple implementations in Python and Mathematica for purpose of testing, or for exploratory studies of methods that make use of pfaffians.

Keywords: Skew symmetric matrices, Pfaffian

PROGRAM SUMMARY

Manuscript Title: Numeric and symbolic evaluation of the pfaffian of general skew-symmetric matrices
Authors: C. Gonzalez-Ballestero, L.M.Robledo and G.F. Bertsch
Program Title: Pfaffian
Journal Reference:
Catalogue identifier:
Licensing provisions:
Programming language: Fortran 77 and 90
Computer:
Operating system:
RAM: bytes
Number of processors used:

*Corresponding author.
E-mail address: luis.robledo@uam.es

Supplementary material:
Keywords: Skew symmetric matrices, Pfaffian
Classification: 4.8 Linear Equations and Matrices
External routines/libraries: BLAS
Subprograms used:
*Catalogue identifier of previous version:**
*Journal reference of previous version:**
*Does the new version supersede the previous version?:**
Nature of problem: Evaluation of the Pfaffian of a skew symmetric matrix.
Evaluation of pfaffians arises in a number of physics applications involving fermionic mean field wave functions and their overlaps.
Solution method: Householder tridiagonalization. Aitken's block diagonalization formula.

*Reasons for the new version:**

*Summary of revisions:**

Restrictions:

Unusual features:

Additional comments: Python and Mathematica implementations are provided in the main body of the paper

Running time: Depends on the size of the matrices. For matrices with 100 rows and columns a few miliseconds are required.

1. Introduction

In a number of fields in physics, the formal equations derived from the theory make use of the pfaffian of some skew-symmetric matrix appearing in the theory. For example, the pfaffian arises in the treatment of electronic structure with quantum Monte Carlo methods [1], the description of two-dimensional Ising spin glasses [2], and the evaluation of entropy and its relation to entanglement [3]. Pfaffians occur naturally in field theory and nuclear physics in formalisms based on fermionic coherent states [4, 5, 6, 7]. A recent application is to the overlap of two Hartree Fock Bogoliubov (HFB) product wave functions [8], needed for nuclear structure theory. While there is a simple formula for the pfaffian of a skew-symmetric matrix M in terms of the determinant,

$$\mathrm{pf}(A) = \sqrt{\det(A)} \tag{1}$$

the so-called "sign problem of the overlap" [9] associated with the square root motivates the use of numerical algorithms that evaluate it directly. The most straightforward method, the rule of "expanding in minors" [10], has bad scaling

with the size of the matrix and is prohibitive for large matrices. In this paper we discuss two alternative methods that have the same scaling property as the normal N^3 algorithms for the determinant. The methods are implemented in the FORTRAN 77 and 90 subroutines provided in the accompanying program library. We also comment on the practical implementation of the two methods in Mathematica and in the Python programming language.

2. Evaluation of the Pfaffian

The Pfaffian $\mathrm{pf}(A)$ is reduced to a simple form that is easily evaluated by making repeated use of transformation formula given in Appendix A,

$$\mathrm{pf}(B^t AB) = \det(\mathrm{B})\mathrm{pf}(A). \tag{2}$$

In order to perform the numerical evaluation of the Pfaffian of a complex skew-symmetric matrix A we reduce the skew-symmetric matrix to a tridiagonal form A_{TR} by using unitary matrices U. Once it is in this form, the evaluation of the pfaffian is straightforward (see below).

2.1. Reduction to tridiagonal form by mean of Householder transformations

In this method, we will use the well-known Householder transformations [11] to reduce A to tridiagonal form. We present it in some detail because the generalization to the complex number field is not entirely trivial.

Complex Householder transformations have the form

$$P = \mathbb{I} - 2\frac{u \otimes u^+}{|u|^2} \tag{3}$$

where u is an arbitrary complex complex row vector $u = (u_1, u_2, \ldots, u_N)$ and $(u \otimes u^+)_{ij} = u_i u_j^*$. The vector u must be chosen to zero all the elements of a vector x except a given one. If we take $u = x \mp e^{i\arg(x_j)}|x|e_j$, with $(e_j)_k = \delta_{jk}$, it can be easily proved that

$$P_u x = \pm e^{i\arg(x_j)}|x|e_j$$

as required. The freedom on the sign in the expression defining the vector u can be used to make sure that the vector u is non zero. The rest of the Householder tridiagonalization procedure follows exactly as in the real case. Consider a skew-symmetric matrix of dimension N (even)

$$A = \left(\begin{array}{c|ccc} 0 & a_{12} & a_{13} & \cdots \\ \hline -a_{12} & & & \\ -a_{13} & & {}^{(N-1)}A & \\ \vdots & & & \end{array}\right) \tag{4}$$

The Householder transformation matrix is

$$P_1 = \left(\begin{array}{c|ccc} 1 & 0 & 0 & \dots \\ \hline 0 & & & \\ 0 & & {}^{(N-1)}P_1 & \\ \vdots & & & \end{array}\right)$$

where ${}^{(N-1)}P_1$ is built by using Eq. (3) and taking the vector x (of dimension N-1) as $(a_{12}, a_{13}, \dots)^T$. The resulting transformed matrix is given by

$$P_1 A P_1^T = \left(\begin{array}{c|ccc} 0 & k_1 & 0 & \dots \\ \hline -k_1 & & & \\ 0 & & {}^{(N-1)}\tilde{A} & \\ \vdots & & & \end{array}\right)$$

where $k_1 = \pm e^{i \arg(a_{12})}|x|$ and the matrix ${}^{(N-1)}\tilde{A}$ is skew-symmetric and given by ${}^{(N-1)}\tilde{A} = {}^{(N-1)}P_1 {}^{(N-1)}A^{(N-1)}P_1^T$. Performing this procedure a total of N-2 times we end up with a tridiagonal and skew-symmetric matrix

$$P_{N-2} \dots P_2 P_1 A P_1^T P_2^T \dots P_{N-2}^T = \begin{pmatrix} 0 & k_1 & 0 & 0 & \dots & 0 \\ -k_1 & 0 & k_2 & 0 & \dots & 0 \\ 0 & -k_2 & 0 & \ddots & \dots & \vdots \\ & & \ddots & 0 & \ddots & \\ 0 & & & \ddots & 0 & k_{N-1} \\ \vdots & & \vdots & & -k_{N-1} & 0 \end{pmatrix} \tag{5}$$

Using now a known property the Pfaffian (see Appendix A) we can deduce from the above identity that $\det(P_1) \dots \det(P_{N-2})\mathrm{pf}(A) = \mathrm{pf}(A_{TR})$ where A_{TR} is the triagonal and skew-symmetric matrix of the right hand side of Eq. (5). Taking into account that the determinant of any Householder matrix is -1 and that N is even, we can express the Pfaffian of A in terms of the pfaffian of the tridiagonal A_{TR}

$$\mathrm{pf}(A) = \mathrm{pf}(A_{TR})$$

As will be shown below the Pfaffian of a tridiagonal skew-symmetric matrix is simply given by $k_1 k_3 \dots k_{N-1} = \prod_{i=1}^{N/2} k_{2i-1}$ (this result can also be obtained using the "minor expansion" formula [10]) and finally we obtain

$$\mathrm{pf}(A) = \prod_{i=1}^{N/2} k_{2i-1}. \tag{6}$$

In terms of numerical stability, the Householder transformation is very robust and there is no need to consider any "pivoting" strategy common to other methods. However, the presence of the square root of x and the argument

$\arg(x_j)$ of complex quantities prevents an easy implementation of the Householder tridiagonalization procedure for symbolic computation. For this purpose the second method described in the next section is far easier to implement.

2.2. Aitken's block diagonalization formula

There is an alternative method for the calculation of the pfaffian, which is also well suited for a symbolic implementation and that relies on an expression for the pfaffian of a bipartite skew-symmetric matrix. Let us start with a general skew-symmetric matrix A (dimension N, even) given by

$$A = \begin{pmatrix} R & Q \\ -Q^T & S \end{pmatrix} \tag{7}$$

where R and S are square skew-symmetric matrices and Q is a general rectangular matrix (to account for the case where R and S have different dimensions). Using Aitken's block diagonalization formula (see [12] for an early use of the formula and [13] for a recent and thorough reference) for a bipartite matrix we obtain

$$\begin{pmatrix} \mathbb{I} & 0 \\ Q^T R^{-1} & \mathbb{I} \end{pmatrix} \begin{pmatrix} R & Q \\ -Q^T & S \end{pmatrix} \begin{pmatrix} \mathbb{I} & -R^{-1}Q \\ 0 & \mathbb{I} \end{pmatrix} = \begin{pmatrix} R & 0 \\ 0 & S + Q^T R^{-1} Q \end{pmatrix} \tag{8}$$

where the matrix $S + Q^T R^{-1} Q$ is referred to in the literature as the Schur complement of the matrix A (see, for instance, [13]). For the special case of a skew-symmetric matrix A, the matrices R and S are also skew-symmetric and the transformation of the matrix A is a congruence (i.e. the matrix acting on the left hand side of A is the transpose of the one acting on the right hand side). Denoting

$$P_1 = \begin{pmatrix} \mathbb{I} & 0 \\ Q^T R^{-1} & \mathbb{I} \end{pmatrix} \tag{9}$$

Eq. (8) becomes

$$P_1 A P_1^T = \begin{pmatrix} R & 0 \\ 0 & S + Q^T R^{-1} Q \end{pmatrix}$$

An equivalent expression involving S^{-1} instead of R^{-1} is easily obtained

$$P_2 A P_2^T = \begin{pmatrix} R + Q S^{-1} Q^T & 0 \\ 0 & S \end{pmatrix}$$

with

$$P_2 = \begin{pmatrix} \mathbb{I} & -Q S^{-1} \\ 0 & \mathbb{I} \end{pmatrix} \tag{10}$$

By using the property $\mathrm{pf}(P^T A P) = \det(P)\mathrm{pf}(A)$ (see Appendix A) and taking into account that $\det P_1 = \det P_2 = 1$, we come to

$$\mathrm{pf}(A) = \mathrm{pf}(R)\mathrm{pf}(S + Q^T R^{-1} Q) \tag{11}$$

$$= \mathrm{pf}(R + Q S^{-1} Q^T)\mathrm{pf}(S) \tag{12}$$

Another nice property of the matrices P_1 and P_2 is that their inverses can be obtained very easily

$$P_1^{-1} = \begin{pmatrix} \mathbb{I} & 0 \\ -Q^T R^{-1} & \mathbb{I} \end{pmatrix} \tag{13}$$

and

$$P_2^{-1} = \begin{pmatrix} \mathbb{I} & QS^{-1} \\ 0 & \mathbb{I} \end{pmatrix} \tag{14}$$

These expressions of the inverses explicitly show that both P_1 and P_2 are not orthogonal matrices.

Let us now apply the above result to an arbitrary skew-symmetric matrix of dimension $N = 2M$ which is written in block form as

$$A = \begin{pmatrix} A^{(1)} & A_{N-1} & A_N \\ -A_{N-1}^T & 0 & a_{N-1,N} \\ -A_N^T & -a_{N-1,N} & 0 \end{pmatrix} \tag{15}$$

where $A^{(1)}$ is a skew-symmetric square matrix of dimension $N-2 = 2(M-1)$ and A_{N-1} and A_N are column vectors $A_{N-1} = \{A_{i,N-1},\, i = 1, N-2\}$ and $A_N = \{A_{i,N},\, i = 1, N-2\}$ both of dimension $(N-2) \times 1$. In the language of Eq (7) the matrix R is the matrix $A^{(1)}$, the matrix Q is a rectangular matrix of dimension $2 \times (N-2)$ made of the two column vectors, A_{N-1} and A_N and finally the matrix S is the 2×2 skew-symmetric matrix with matrix element $S_{12} = a_{N-1,N}$. Using the ideas of Aitken's block diagonalization formula, it is easy to shows that the matrix $\tilde{A} = D_1^T A D_1$ is in block diagonal form

$$\tilde{A} = \begin{pmatrix} \tilde{A}^{(1)} & 0 & 0 \\ 0 & 0 & a_{N-1,N} \\ 0 & -a_{N-1,N} & 0 \end{pmatrix} \tag{16}$$

with a matrix D_1 of the form

$$D_1 = \begin{pmatrix} \mathbb{I}_{N-2} & 0 & 0 \\ X & 1 & 0 \\ Y & 0 & 1 \end{pmatrix} \tag{17}$$

where $\mathbb{I}_{N-2}$ stands for the identity matrix of dimension $N-2$ and both X and Y are row vectors of dimension $1 \times (N-2)$ and given by $X = -a_{N-1,N}^{-1} A_N^T$ and $Y = a_{N-1,N}^{-1} A_{N-1}^T$. In the above equation 16, the skew-symmetric matrix $\tilde{A}^{(1)}$ is given by

$$\tilde{A}^{(1)} = A^{(1)} + A_N (a_{N-1,N})^{-1} A_{N-1}^T - A_{N-1} a_{N-1,N}^{-1} A_N^T \tag{18}$$

Taking into account that $\det D_1 = 1$ then $\mathrm{pf}(A) = \mathrm{pf}(\tilde{A}) = a_{N-1.N}\mathrm{pf}(\tilde{A}^{(1)})$. The algorithm can be applied recursively to $\tilde{A}^{(1)}$ to obtain
$\mathrm{pf}(A) = a_{N-1,N}\tilde{a}^{(1)}_{N-3,N-2}\mathrm{pf}(\tilde{A}^{(2)})$ so as to reduce, after $M-1$ iterations, the computation of the pfaffian to the product of the corresponding elements.

This procedure can be easily implemented for a skew-symmetric tridiagonal matrix, as the transformed matrices in Eq (18) coincide with the original ones; for instance, $\tilde{A}^{(1)} = A^{(1)}$. As a consequence, the pfaffian of a tridiagonal matrix is given by

$$\mathrm{pf}\begin{pmatrix} 0 & d_1 & 0 & 0 & \dots & 0 \\ -d_1 & 0 & d_2 & 0 & \dots & 0 \\ 0 & -d_2 & 0 & \ddots & \dots & \vdots \\ 0 & 0 & \ddots & 0 & \ddots & 0 \\ \vdots & \vdots & \vdots & \ddots & 0 & d_{2N-1} \\ 0 & 0 & \cdots & 0 & -d_{2N-1} & 0 \end{pmatrix} = d_1 d_3 \dots d_{2N-1} = \prod_{i=1}^{N} d_{2i-1}$$

2.2.1. Pivoting

As a consequence of the division by matrix elements like $a_{N-1,N}$ in the first iteration, the numerical stability of the algorithm requires the use of pivoting strategy in the implementation of the method. Full pivoting amounts to search the whole matrix for the matrix element with the largest modulus and exchange it with the required matrix element. For instance, in the first iteration of the procedure, the matrix element $a_{p,q}$ $(p < q)$ with the largest modulus is searched for and exchanged with the matrix element $a_{N-1,N}$. In this way we avoid dangerous divisions by small (or even zero) matrix elements. We have to take into account that in the present case, the exchange of both columns and rows is required to preserve the skew-symmetric nature of the matrices involved. To carry out the exchange of rows and columns we will use the exchange matrix $P(ij)$ that, when applied to the right of an arbitrary matrix, exchanges columns i and j. The exchange matrix is given by the matrix elements

$$P(ij)_{kl} \quad = \quad \delta_{kl} - \delta_{i,l}\delta_{i,k} - \delta_{j,l}\delta_{j,k} + \delta_{i,l}\delta_{j,k} + \delta_{j,l}\delta_{i,k}. \tag{19}$$

To exchange the corresponding rows we have to apply $P(ij)^T$ to the left of the matrix (notice that $P(ij)$ is symmetric). With the help of these matrices we can write the matrix after pivoting $a_{p,q}$ with $a_{N-1,N}$ (and $a_{q,p}$ with $a_{N,N-1}$) as

$$A_P = P^T(N-1,p)P^T(N,q)\, A\, P(N-1,p)P(N,q)$$

As a consequence of such exchange and taking into account that $\det P(ij) = -1$ we can conclude that the pfaffian of A does not change by the pivoting procedure. Finally we obtain

$$A = P(N,q)P(N-1,p)\, A_P\, P^T(N-1,p)P^T(N,q)$$

$$= P(N,q)P(N-1,p){D_1^T}^{-1}\tilde{A}_P D_1^{-1} P^T(N-1,p)P^T(N,q)$$

where $\tilde{A}_P$ has the same structure as $\tilde{S}$ in Eq. (16). As before, $\mathrm{pf}(A) = \mathrm{pf}(\tilde{A}_P) = (A_P)_{N-1,N}\,\mathrm{pf}(\tilde{A}_P^{(1)})$ and repeating recursively the whole procedure $M-1$ times we obtain the pfaffian as the product of the corresponding matrix elements.

2.2.2. Cholesky like decomposition of a skew-symmetric matrix

Although it is not necessary in order to compute the pfaffian, it can be useful to show that even with pivoting we can write the matrix A as

$$A = PL^T \tilde{A} LP \tag{20}$$

where P is the product of exchange matrices as in Eq (19), L is the product of matrices of the D^{-1} type, Eq (17), and therefore is a lower triangular matrix with ones in the main diagonal and finally, $\tilde{A}$ is a skew-symmetric matrix in canonical form, i.e. a block diagonal matrix with skew-symmetric, 2×2 blocks in the diagonal. This decomposition of a general skew-symmetric matrix A resembles the Cholesky decomposition of a general matrix and can be useful in formal manipulations like, for instance, the inversion of the matrix A. In order to show that Eq (20) holds the only required property is that, when applying the pivoting procedure to $\tilde{A}_P^{(1)}$ the exchange matrices required $P(N-2,s)P(N-3,r)$ have the property of preserving the structure of the matrix D_1(and its inverse). For instance,

$$D_1^{T\,-1} P(N-2,s)P(N-3,r) = P(N-2,s)P(N-3,r)\tilde{D}_1^{T\,-1}$$

with $\tilde{D}_1^{T\,-1}$ a matrix that is obtained from $D_1^{T\,-1}$ by exchanging rows $N-2$ and s and rows $N-3$ and r and therefore has the same upper triangular structure with ones in the diagonal as the original matrix $D_1^{T\,-1}$. Using this property we can move all the exchange matrices to the right (or to the left) and the remaining matrix will be the product of triangular matrices (lower for products involving D^{-1}) with ones in the diagonal.

As mentioned earlier, Aitken's method is better adapted to symbolic evaluations. However, one must take care that in each step of the process some specific matrix elements are non-zero.

3. Fortran implementation

The implementation of the algorithms considered in this paper in a high level computer language is straightforward. However, specific code in FORTRAN (both 77 and 90, real and complex arithmetic) is provided along with this paper. The algorithms are easy to follow and the comments included in the code are useful guides. Just a few comments are in order: to implement the tridiagonalization procedure in Fortran, it is advantageous to use the BLAS package [14] to perform the required matrix by vector multiplication and rank two update. Unfortunately there are no equivalent in the skew-symmetric case of the routines SYM (to multiply a symmetric matrix by a vector) or SYR2 (to perform a symmetric rank two update) but the general procedures GEMV and GERU can be used instead. On output, both the pfaffian of the matrix and the set of vectors required to bring it to tridiagonal form are returned. For the implementation of the method based on Aitken's block diagonalization formula a pivoting strategy is required. We have used full pivoting in our implementation due to its robustness. The routines provided only require the upper

part of the skew-symmetric matrix. The lower part is destroyed and replaced with the tridiagonal transformation matrix that brings the skew-symmetric matrix to canonical form upon congruence. An integer vector is also returned to reconstruct the required exchange of rows and columns.

Perhaps the best test to check the validity of the two implementations is to compute the pfaffian of a skew-symmetric matrix using both procedures in order to compare the output. If it is the same up to a given accuracy then it is very likely that the two implementations are correct. We have writen a test program (also included in the distribution) that generates skew-symmetric matrices of given dimension with random entries and compute the pfaffian using both techniques. In our tests the pfaffians computed both ways coincide up to one part in 10^{10} with dimensions of the matrices of one thousand. This result also supports the adequacy of the implementation in terms of numerical stability. Another possibility to test the numerical implementation is to use the analytical formula given in Appendix B for a specific kind of 8×8 matrices. A test program implementing this approach has also been included in the distribution.

To finish this section we will briefly comment on the timing of the FORTRAN numerical implementations mentioned. In a modern personal computer under Linux the computation of the pfaffian of a 100×100 matrix takes a few milliseconds in both implementations and the timing scales roughly as the cube of the dimension of the matrix in such a way that for matrices of a 1000×1000 dimension the time is of the order of a few seconds.

4. A simple Python implementation

We provide here a simple implementation of the tridiagonal reduction method (see [12] and [15]) in Python, which may be useful for testing purposes. It is similar to the Householder, but it only use simple row and column operations that have determinants of unity. The code is:

```
from numpy import *
def pfaff_py(m) :
  mat=copy(m)
  ndim = shape(mat)[0]
  t1=1.0
  for j in range(ndim/2) :
    t1 *= mat[0,1]
    print 't1', t1
    if j <ndim/2-1 :
      ndimr=shape(mat)[0]
      for i in range(2,ndimr) :
        if mat[0,1] != 0.0 :
          tv=mat[1,:]*mat[i,0]/mat[1,0]
          mat[i,: ] -= tv
          tv=mat[:,1]*mat[0,i]/mat[0,1]
          mat[:,i ] -= tv
```

```
        else :
          print 'need to pivot'
          raise Exception
      mat=mat[2:,2:]
  return t1
```

The user should be cautioned that the algorithm is not guaranteed to be stable without an additional pivot step. Also, the matrix is assumed to have been constructed with the `array` function in the Numpy library.

5. A simple Mathematica implementation

We also provide a simple Mathematica implementation of the method based on Aitken's block diagonalization formula. As mentioned above, this method requires pivoting to avoid divisions by small (or zero) numbers. In the symbolic implementation, this issue is solved by replacing the denominator by a variable (OO on the implementation below) in case it is zero and an additional limit when the variable tends to zero is performed at the end. The two Mathematica modules required are:

```
Aitken[M_,n_,OO_]:=
Module[{MM=M,i,j,p},
If[MM[[n-1,n]]==0,MM[[n-1,n]]=OO;MM[[n,n-1]]=-OO];
p=MM[[n-1,n]];
For[i=1,i<=n-2,i++,
   For[j=1,j<=n-2,j++,
      MM[[i,j]]=M[[i,j]]+(MM[[i,n]]*MM[[j,n-1]]-MM[[j,n]]*MM[[i,n-1]])/p
      ]
   ];
 MM];

pfaffian[S_]:=
Module[{T=S,n,p},
n=Length[T]/2;
If[T[[2*n-1,2*n]]==0,T[[2*n-1,2*n]]=OO;T[[2*n,2*n-1]]=-OO];
For[n=Length[T]/2;p=T[[2*n-1,2*n]],n>1,n--,
   T=Aitken[T,2*n,OO];
   p=p*T[[2*(n-1)-1,2*(n-1)]]
   ];
Limit[p,OO->0]];
```

6. Conclusions

The issue of how to compute both numerically and symbolically the pfaffian of a skew-symmetric matrix has been addressed using two different approaches.

Numerical stability issues are discussed and methods to assure the desired accuracy are fully incorporated. A collection of subroutines and test programs in FORTRAN (both 77 and 90, double precision and complex) are provided. A few comments on the implementation of the algorithms in Mathematica and Python are also given.

7. Acknowledgements

We acknowledge K. Roche for a careful reading of the manuscript and several suggestions. This work was supported by MICINN (Spain) under research grants FPA2009-08958, and FIS2009-07277, as well as by Consolider-Ingenio 2010 Programs CPAN CSD2007-00042 and MULTIDARK CSD2009-00064.

Appendix A. Definition and basic properties of the pfaffian

The pfaffian of a skew-symmetric matrix R of dimension $2N$ and with matrix elements r_{ij} is defined as

$$\mathrm{pf}(R) = \frac{1}{2^n}\frac{1}{n!}\sum_{\mathrm{Perm}} \epsilon(P) r_{i_1 i_2} r_{i_3 i_4} r_{i_5 i_6} \dots r_{2n-1,2n}$$

where the sum extends to all possible permutations of $i_1, \dots, i_{2n}$ and $\epsilon(P)$ is the parity of the permutation. For matrices of odd dimension the pfaffian is by definition equal to zero. As an example, the pfaffian of a 2×2 matrix R is $\mathrm{pf}(R) = r_{12}$ and for a 4×4 one $\mathrm{pf}(R) = r_{12}r_{34} - r_{13}r_{24} + r_{14}r_{23}$. Useful properties of the pfaffian are

$$\mathrm{pf}(P^T R P) = \det(P)\mathrm{pf}(R), \tag{A.1}$$

$$\mathrm{pf}\begin{pmatrix} 0 & R \\ -R^T & 0 \end{pmatrix} = (-1)^{N(N-1)/2}\det(R)$$

where the matrix R is $N \times N$ and

$$\mathrm{pf}\begin{pmatrix} R_1 & 0 \\ 0 & R_2 \end{pmatrix} = \mathrm{pf}(R_1)\mathrm{pf}(R_2)$$

where R_1 and R_2 are skew-symmetric matrices. The matrices may be defined on the real or on the complex number fields.

Appendix B. Pfaffian of a test matrix

In this appendix we give the expression of the pfaffian of a test matrix which is big enough as not to be trivial but on the other hand is small enough as to render the explicit expression of the pfaffian manageable. The expression given

below can be used to check both numerical and symbolic implementations of the pfaffian.

Consider the two general skew-symmetric matrices of dimension 4

$$M = \begin{pmatrix} 0 & f_1 & m_{11} & m_{12} \\ -f_1 & 0 & m_{21} & m_{22} \\ -m_{11} & -m_{21} & 0 & f_2 \\ -m_{12} & -m_{22} & -f_2 & 0 \end{pmatrix}$$

and

$$N = \begin{pmatrix} 0 & g_1 & n_{11} & n_{12} \\ -g_1 & 0 & n_{21} & n_{22} \\ -n_{11} & -n_{21} & 0 & g_2 \\ -n_{12} & -n_{22} & -g_2 & 0 \end{pmatrix}$$

where the matrix elements can be complex numbers. With these two matrices and the identity 4×4 matrix we build the skew-symmetric matrix

$$S = \begin{pmatrix} N & -\mathbb{I} \\ \mathbb{I} & -M^* \end{pmatrix}$$

of dimension 8×8 (see Ref [8] for the physical context of this matrix). It is relatively easy to compute its pfaffian

$$\text{pf}[S] = 1 + f_1^* g_1 + f_2^* g_2 + m_{11}^* n_{11} + m_{22}^* n_{22} + m_{12}^* n_{12} + m_{21}^* n_{21} +$$

$$+ (f_1^* f_2^* - m_{11}^* m_{22}^* + m_{12}^* m_{21}^*)(g_1 g_2 - n_{11} n_{22} + n_{12} n_{21})$$

References

[1] M. Bajdich, L. Mitas and L.K. Wagner, Phys. Rev B77, 115112 (2008)

[2] C.K. Thomas and A. A. Middleton, Phys. Rev E80, 046708 (2009)

[3] J.-M. Stéphan, S. Furukawa, G. Misguich, and V. Pasquier, Phys. Rev B80, 184421 (2009)

[4] F.A. Berezin, *The Method of Second Quantization* (Academic Press, New York, 1966)

[5] Y. Ohnuki, and T. Kashiwa, Prog. Theor. Phys. 60, 548 (1978).

[6] John R. Klauder, Bo-Sture Skagerstam, *Coherent states: applications in physics and mathematical physics* (World Scientific, Singapore, 1985)

[7] G.H. Lang, C.W. Johnson, S.E. Koonin, and W.E. Ormand, Phys. Rev. **C**48, 1518 (1993)

[8] L.M. Robledo, Phys. Rev. C79, 021302 (2009)

[9] K. Neergard, and E. Wüst, Nucl. Phys. **A**402, 311 (1983)

[10] E.R. Caianiello, *Combinatorics and renormalization in Quantum Field Theory* (W.A. Benjamin, Massachusetts, 1973)

[11] G. H. Golub and C. F. Van Loan, *Matrix Computations* (Johns Hopkins University Press, Baltimore, 1996).

[12] J. R. Bunch, Math of Comp. 38, 475 (1982)

[13] F. Zhang Ed., *The Schur Complement and Its Applications (Numerical Methods and Algorithms)* (Springer, Berlin, 2005)

[14] J. J. Dongarra, J. Du Croz, S. Hammarling, and R. J. Hanson, ACM Trans. Math. Soft. 14, 1 (1988),

[15] J. O. Aasen, BIT 11, 233 (1971)